HUMAN COMPUTER INTERFACE BASED ON HAND GESTURE RECOGNITION

A Thesis
Presented to
The Academic Faculty

by

Arnaud J. Bernard

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2010

HUMAN COMPUTER INTERFACE BASED ON HAND GESTURE RECOGNITION

Approved by:

Benny K. Bing, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Gee-Kung Chang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Biin Hwang (Fred) Juang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: August 23, 2010

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

Blob	Binary Large OBject, group of connected pixels.
DFT	Discrete Fourier Transform.
FFT	Fast Fourier Transform.
Fps	Frames per second.
H.264	Latest MPEG video codec.
HCI	Human Computer Interaction.
HDTV	High-definition television.
MB	Macroblock, small subdivision of a frame, 16x16 pixels in H.264.
MPEG	Moving Picture Experts Group.
MV	Motion Vector.
OpenCV	Open Computer Vision library, open source C/C++ library initially developed by Intel now hosted by WillowGarage: `http://opencv.willowgarage.com/`.
RGB	Red, Green, Blue color space.
ROI	Region Of Interest, part of a frame we want to process.
SAD	Sum of Absolute Differences.
TV	Television.
Webcam	Video capture device connected to a computer or computer network.
YC_rC_b	YC_rC_b color space (also called YUV), Y stands for luma, C_r stands for red chroma and C_b stands for blue chroma.

SUMMARY

After introducing the subject matter in Chapter 1, a literature survey is covered in Chapter 2. In Chapter 3, we describe existing computer vision methods such as skin color detection and Haar cascade of features when applied to hand detection. We then describe new methods based on template matching and H.264 motion vectors. The performance of these methods are compared in terms of computational efficiency and accuracy. Different methods for trajectory analysis are presented and compared in terms of responsiveness. In Chapter 4, we describe a simple method for hand reference acquisition using a 3D stereo webcam. A method to estimate a threshold for a depth map is proposed to detect hand gestures without any reference. In addition, a technique based on edge detection is proposed to estimate 3D hand rotation. Finally, in Chapter 5, a framework for extended features such as zoom and two hand gesture recognition is presented. We also describe different human computer interfaces for video browsing and gaming applications.

Hand gesture recognition has a wide range of applications, including video browsing on a TV and gaming. This thesis presents different methods to perform hand gesture recognition using a single webcam or a stereo webcam.

After introducing the subject matter in Chapter 1, a literature survey is covered in Chapter 2. In Chapter 3, we describe existing computer vision methods such as skin color detection and Haar cascade of features when applied to hand detection. We then describe new methods based on template matching and H.264 motion vectors.The performance of these methods are compared in terms of computational efficiency and accuracy. Different methods for trajectory analysis are presented and compared in terms of responsiveness.

In Chapter 4, we describe a simple method for hand reference acquisition using a 3D stereo webcam. A method to estimate a threshold for a depth map is proposed to detect hand gestures without any reference. In addition, a technique based on edge detection is

proposed to estimate 3D hand rotation.

Finally, in Chapter 5, a framework for extended features such as zoom and two hand gesture recognition is presented. We also describe different human computer interfaces for video browsing and gaming applications.

Two papers describing our work have been accepted or published in the following conferences:

- A. Bernard, B. Bing, "Hand Gesture Video Browsing for Broadband-Enabled HDTVs", *IEEE Sarnoff Symposium*, April 12-14 2010, New Jersey, USA

- A. Bernard, B. Bing, "Automatic Hand Reference Acquisition Using a Stereo 3D Webcam", *ACM/IEEE International Conference on Distributed Smart Cameras*, Aug. 31-Sept. 14 2010, Georgia, USA

CHAPTER I

INTRODUCTION

With the improvement of multimedia technologies such as broadband-enabled HDTV, video on demand and internet TV, the computer and the TV are merging to become a single device. Moreover the previously cited technologies as well as DVD or Blu-ray can provide menu navigation and interactive content.

The growing interest in video conferencing led to the integration of the Webcam in different devices such as laptop, cell phones and even the TV set. Our approach is to directly use an embedded webcam to remotely control a TV set using hand gestures. Using specific gestures, a user is able to control the TV. A dedicated interface can then be used to select a TV channel, adjust volume or browse videos from an online streaming server.

This approach leads to several challenges. The first is the use of a simple webcam which leads to a vision based system. From the single webcam, we need to recognize the hand and identify its gesture or trajectory. A TV set is usually installed in a living room which implies constraints such as a potentially moving background and luminance change. These issues will be further discussed as well as the methods developed to resolve them. Video browsing is one example of the use of gesture recognition. To illustrate another application, we developed a simple game controlled by hand gestures.

The emergence of 3D TVs is allowing the development of 3D video conferencing. Therefore we also consider the use of a stereo camera to recognize hand gesture.

CHAPTER II

PREVIOUS WORK

2.1 Application Domains

Work on hand gesture recognition has been reported in several fields of research. A major interest is the interpretation of sign language [27, 11]. The automatic recognition of sign language gesture is the first step in the creation of an automatic translator between sign language and spoken language. However, sign language features a lot of complex gestures and only a restricted set is addressed. Another research focus is using hand gestures for human-robot interactions [8]. Hand gestures can be directly used to control the robot. For example a robot arm can be coupled to a human arm using hand gesture recognition [31]. Hand gestures can also be used to characterize human activity and let robots adjust their behavior to help humans [26]. Another approach is to communicate with the robot using hand gestures; a robot interpreting hand gestures to get directions as been reported [19]. Other various interactions can be interpreted from hand gestures such as the leading of a musical band [15]. A detailed review of hand gestures capabilities has been proposed by Pavlovic et al. [25].

Concerning the interaction between a user and a TV set, a few contributions have been reported. Freeman et al. [7] propose to use a computer wired to a remote control, the computer is used to do gesture recognition and send commands to the TV using the remote control. Their approach require an independent display connected to the computer to provide feedback to the user. The use of two different screens prevents direct manipulation. Zaletelj et al. [35] mention a TV setup for desk interfaces and augmented reality.

2.2 Hand Gesture Recognition

2.2.1 Hand Segmentation

Different approaches have been used to detect a user's hand using video data. One of the most common methods is to use colored gloves to identify the hands [16]. However, if the

user has to wear colored gloves to use our system, it will degrade the simplicity of the human-TV interaction.

A simple way of describing a complex scene is to use background modeling and extraction. Static or slowly adaptive background modeling [9] requires the scene to be empty at the beginning of the tracking which is not likely to be the case for a living room environment. For example, someone may stand in front when the TV is turned on. Adaptive background models [33] perform better and can be used to the select areas of interest in the image to speed up processing, but still suffer from the same problem described previously.

Color detection is a common method to detect the hand [1, 21, 4, 7, 20, 32]. This method works in specifics situations and requires either extracting the skin color of the hand or making assumptions about it [21]. Making loose assumptions [20] on skin colors (for example simple threshold of RGB components) may result in false positives, hence other techniques must be used to improve the reliability of hand recognition. In addition, extraction of the skin color must be performed regularly as it is sensitive to indoor lighting variation.

Another approach is to detect edges is to use either a Sobel filter or a Canny filter, and then extract a hand shape from these edges [4]. This method appears to be efficient on a clear background. Nonetheless, these filters are sensitive to detected edges and in particular, to edges located in the background. This is why this technique is often combined with background modeling and skin color detection [11].

Yet another method to remove background information is to use a pattern matching methods to search for a pattern as defined by a set of references.Since this method may take up significant processing time, a decision tree using small Haar-like features and Adaboost method have been suggested [34, 8]. Pattern matching methods require a very large database and a long training time. In order not to rely on a large hand gesture database, we prefer to use simpler pattern matching method based on the user's hand reference.

As in a living room situation, the background can be transformed into non-moving objects. MPEG encoded videos provide a simple way to segment moving objects. In particular, the H.264 video coding standard uses motion vectors which can be used to track

moving objects in a video [18].

Another common way to track hand gestures is to use two or more cameras. The cameras can be placed in front or on the side [16] to be able to precisely triangulate the 3D position of a moving hand. A stereo camera [15], which consists of two cameras located side by side, can provide depth information to simplify hand recognition. Although these methods achieve good results, the need for multiple cameras may not be convenient.

2.2.2 Trajectory Analysis

Trajectory analysis can be performed by many ways, either by simple matching such as features points or direction [26]. More complex modeling theories have emerged to recognize complex gesture trajectories using Hidden Markov Models (HMM) [4]. HMM have the advantage that the states can be mapped directly with part of the hand trajectory. However these methods require a lot of trajectory samples for training and therefore are difficult to adapt to user-specific gestures and create extra steps for simple applications such as video browsing.

Two-hand trajectories have been studied to handle more complex gestures [14, 27]. These studies are focused on the recognition of gestures using two visible hands at the same time but assume no occlusion or the temporary loss of tracking of one hand. We propose a scheme to handle both single handed gestures and two handed gesture at the same time. Another issue for trajectory analysis is the temporal segmentation [1]. Although it is not often mentioned, the recognition accuracy relies on the capacity to determine the start and the end of the gesture [25]. We propose to use a different hand static gesture to signal the end of the trajectory to solve this problem.

2.2.3 Feedback

Lin et al. [20] described three applications for their hand gesture recognition system with feedback. The first consists of controlling the mouse cursor of a computer. Although they demonstrate the feasibility of such an interface, the method suffers from a lack of precision with this method. The second application is the animation of a 3D model of the hand, which is similar to controlling a robotic hand [31] but with potential applications in virtual reality

environments. The last application is a virtual DJ system which allows the use to control a virtual mixer-board. Chai et al. use hand gestures to control a picture gallery browsing interface [3]. Gestures are matched to actions such as opening and closing or directional commands. We propose to use different methods of feedback based on direct manipulation of objects such as the manipulation of videos metaphors for video browsing, a mouse cursor for web browsing and the manipulation of games elements for video games.

2.3 Quality Criteria

Hand gestures recognition accuracy can be evaluated using different criteria. The number of accurate detections versus the number of false positives is considered the most important [25]. In addition, the processing time is often considered [25]. A reasonable frame rate for real time analysis is between 10 Fps and 30 fps. The proposed methods aim at achieving this frame rate on a standard computer.

CHAPTER III

SINGLE WEBCAM HAND GESTURE RECOGNITION

We wish to recognize gestures on a complex background such as the living room which can be surrounded by moving persons and objects in the background. The webcam is either embedded in the screen of the computer or placed on top with the user facing it. Our method is experimented on 640x480 videos acquired with different webcams. Our goal is to process at 25 fps and perform real time processing using a standard personal computer. The programming was done in C/C++ using the OpenCV library[1].

3.1 Techniques for Hand Segmentation

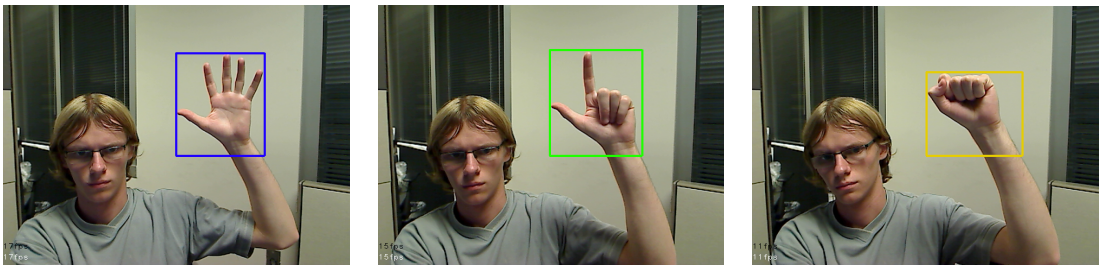
We want to recognize specific gestures. To do that we process frames to locate the hand and its gesture. Figure 1 presents some hand shape examples successfully matched by one of the described algorithms.

3.1.1 Color Based

3.1.1.1 Principle

As discussed in the previous chapter, color based segmentation is the most commonly used technique and is simple to use and computational efficient. Our method extracts pixels of the skin color and attempts to recognize a hand gesture from the contours. Figure 2

[1]OpenCV wiki and documentation: `http://opencv.willowgarage.com/wiki/`



(a) Palm gesture (b) Finger gesture (c) Fist gesture

Figure 1: Hand shape examples.

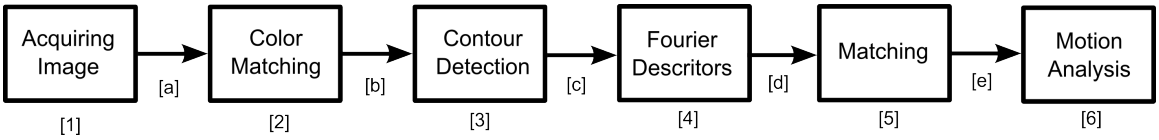


Figure 2: Flowchart of the skin color method.

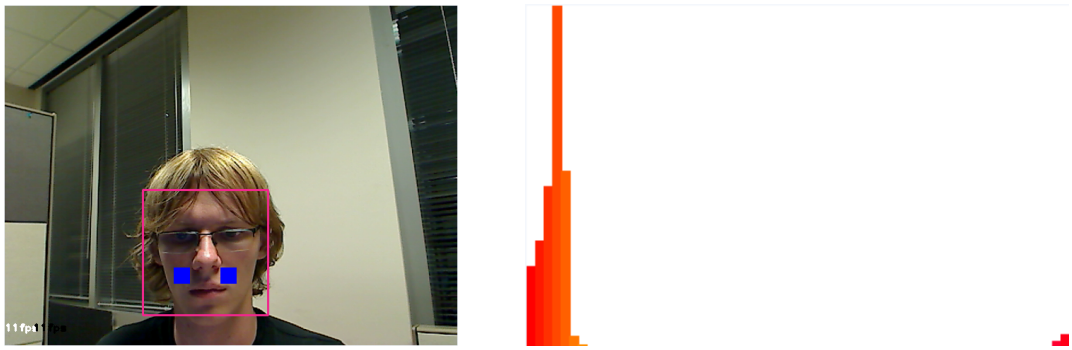
presents a flowchart for this method.

3.1.1.2 Skin Color Extraction

The color based segmentation relies on the identification of skin colored pixels in a captured video frame. These pixels are grouped by regions and extracted as Blobs. These blobs are then processed to identify if they represent the hand.

The identification of skin colored pixel relies on the sampling of one's skin color. To extract the skin color attributes, either loose assumptions can be made in RGB color space or sampling can be done [17]. Since loose assumptions may not adapt to different lighting, we chose a sampling method. This sampling can be done by different ways. The easiest way is to use another segmentation method to get a picture of the hand and sample pixel values from it. A specific framework to perform this is detailed in Subsection 4.2.2. Another efficient method is to extract color from the user's skin. This will allow different color skin types to be detected. As Viola et al. [34] provided an efficient way of detecting a face, we can first locate the head and then retrieve the skin color from the user's cheeks.

The RGB color format does not provide good samples since it is well known that the RGB color format does not match well to the human visual system. The YC_rC_b color format is known for matching the perception of color and is therefore used in video coding In the HSV (define) color space, the hue represents the color, the grayscale saturation, and the value the luminance component. Therefore, only one parameter is used to characterize a color. This allows to direct manipulation of the hue component of a video frame. The drawback is that the hue component of wood colors is close to the hue component of skin color, which affects the reliability of this method. To sample the skin color, we adopt an histogram representation of the color. The height of a histogram column represents the proportion of pixel having this hue in the skin color sample. Figure 3(a) shows the use of



(a) Head skin sampling (b) Resulting Hue Histogram

Figure 3: Skin color extraction using head detection.

(a) Frame with detected hand (b) Backprojection

Figure 4: Skin color backprojection result.

the head detection method to sample the skin color where the pink square is the result of the head detection and the blue squares represent the regions from where the skin color was extracted. Figure 3(b) presents the histogram resulting from this sample.

Using this histogram, we want to detect the parts of the video frame which belong to the user skin. To do that, we use a method called backprojection. This operation affects the histogram value of each pixel. Therefore the brighter a pixel becomes after backprojection, the higher chances are that it belongs to a skin region. Figure 4 shows an example of a frame resulting from a backprojection. We can see that this method effectively differentiate skin pixel from others.

3.1.1.3 Hand Recognition

We retrieve contours from the binary mask obtained and discard very small contours to avoid false positives. We only take the upper part of the binary object so that exposed arm will not interfere. We set a maximum ratio for the height of the object on the width of the object to 1.1, which corresponds to the normal ratio for the palm of a hand. We first distribute the contour's points at regular intervals along the contour, and then we transform the resulting vector of points in the frequency domain using Fourier's Descriptors. Fourier Descriptors provide a simple way to obtain a translation invariant and scale invariant contour. Eq. (1) shows the computation of the Fourier's descriptors.

$$FD(\omega) = \sum_{k=0}^{N-1} \left(Pt_k.x + j \times Pt_k.y\right) e^{j2\pi\left(\frac{\omega k}{N}\right)} \tag{1}$$

Where Pt_k is the k^{th} point of the contour, N is the number of points in the contour and ω is the frequency. We set the first coefficient of Fourier's Descriptor to zero to have a translation invariant pattern and we divide each coefficient by the second coefficient so that our pattern becomes scale invariant. We apply a low pass filter and calculate the distance between the contour and a reference and then perform the inverse transform. We use Eq. (2) to compute the distance between the reference and a contour retrieved from the image.

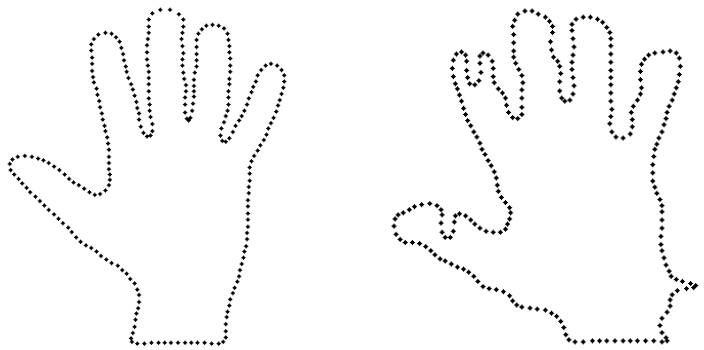
$$Dist = \sum_{k=0}^{N-1} \sqrt{\left(Pt_k.x - Ref_k.x\right)^2 + \left(Pt_k.y - Ref_k.y\right)^2} \tag{2}$$

Where Ref_k is the k^{th} point of the reference contour. Figure 5(a) presents an example of reference contour and Figure 5(b) presents an actual detected contour matched with the reference.

3.1.2 Learning-based Gesture Recognition

3.1.2.1 Principle

Viola et al. [34] have described a very efficient way to recognize specific objects in grayscale images. They provide a supervised learning method based on Adaboost and Haar-like



(a) Reference contour (b) Detected contour

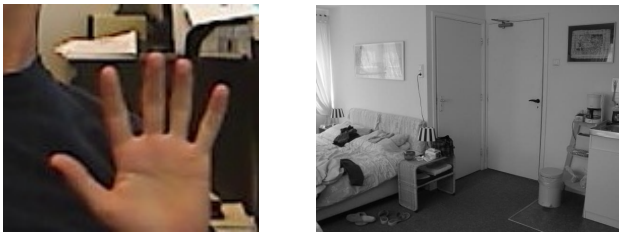
Figure 5: Fourier descriptors of two hands.

features which is proven to efficiently recognize faces. The OpenCV distribution comes with a trained frontal face detector [2] which we used in Subsection 3.1.1 to sample the skin color from the cheeks. The algorithm is designed to work well for rigid objects and characteristic views, which is our case. The user is standing in front of the TV. If we consider one gesture at a time, for a specific gesture, the fingers can be static and the hand considered as a rigid object.

3.1.2.2 Supervised Learning

The training requires a large database of positive and negative frames. Positive frames are frames containing a hand to be recognized while negative frames are frames without any hand reference. The steps to train the Haar cascade of features[2] include manually segmenting the positive samples, converting samples to a uniform set and starting the learning process. To create our positive data set, we used 4 different webcams. Part of the image segmentation was done automatically using the previous methods and the rest was done manually to avoid a bias due to the use of one of our method. The negative samples were taken from the Haar training google code tutorial [3]. The method requires a number of sample to be in the order of thousands. We used 952 positive samples and 1602 negative samples to train a Haar cascade to recognize a palm gesture. Figure 6(a) presents an example of a positive sample and Figure 6(b) presents an example of a negative sample.

[2]The Haar cascade training procedure: `http://note.sonots.com/SciSoftware/haartraining.html`
[3]`http://code.google.com/p/tutorial-haartraining/`



(a) Positive sample (b) Negative sample

Figure 6: Examples of samples used to train the haar cascade.

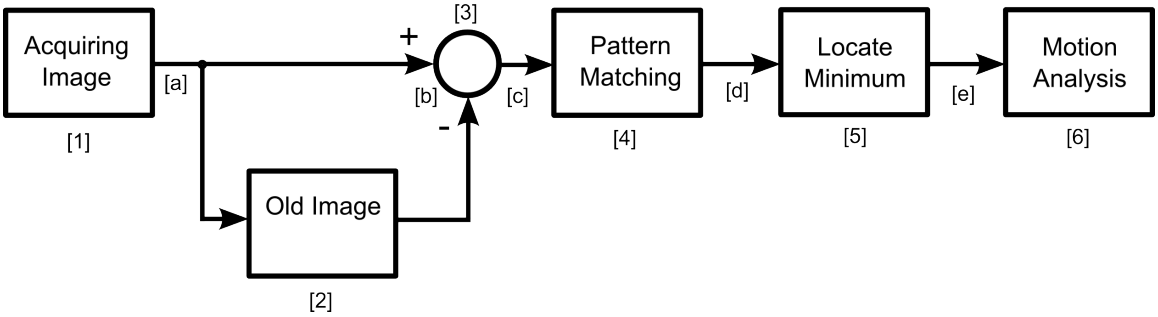


Figure 7: Flowchart of the template matching method.

3.1.3 Template Matching

3.1.3.1 Principle

This method computes the difference between two successive video frames, which is usually stable enough to enable hand recognition using to a reference. It has the advantage of being computationally efficient for a fewer references. Figure 7 presents the flowchart of this method.

3.1.3.2 Description

We first compute the difference between the image and the previous one, and convert it to grayscale. The grayscale map is obtained by adding the contribution of each channel of the RGB original frame to the difference image. Then, we established a distance map between a reference and the motion observed with the webcam. The map is obtained by differencing the grayscale image and the reference throughout the grayscale image. The Sum of Absolute Difference (SAD) described in Eq. (3) is used to create the correlation

(a) Reference template for a finger gesture

(b) Reference template for a palm gesture

Figure 8: Reference templates for the hand template matching method.

map.

$$\begin{aligned} Dist_{Img \leftrightarrow Ref}(x,y) &= \sum_{x'=0}^{m} \sum_{y'=0}^{n} \left(Img\left(x+x', y+y'\right) - Ref\left(x'+y'\right) \right)^2 \\ 0 &\leq x \leq M-m-1 \\ 0 &\leq y \leq N-n-1 \end{aligned} \tag{3}$$

Where Img is the residual image resulting from the difference of two consecutive frames, Ref is the reference, (x, y) are the coordinates of the destination and (x', y') the coordinates of the reference, M and N are the width and height of Img and m and n are the width and height of Ref. We obtain a residual map of size $(M - m - 1) \times (N - n - 1)$. We select the maximum correlation (minimum residual) and compare it to a threshold. If the threshold is reached, we can accurately predict that the hand is at the estimated position. The reference from the user must be acquired off-line. Figure 8 presents two examples of the patterns to be matched and a difference (residual) frame.

The threshold is determined using the following procedure. The distance between a reference and itself is computed using Eq. (3) where $Img = Ref$. We obtain a single value since $M = m$ and $N = n$. For each the threshold is set as a fixed percentage of this value as shown in Eq. (4).

$$Treshold = \alpha \times Dist_{Ref \leftrightarrow Ref}(0,0) \tag{4}$$

Table 1: Processing time for different downsampling range and amount of scaling.

	Downsampling range					
Amount of scaling	1	2	3	4	5	6
1	47.1ms	11.6ms	5.41ms	2.89ms	2.02ms	1.31ms
2	95.3ms	22.9ms	11.5ms	6.56ms	3.53ms	2.61ms
4	192ms	46.6ms	22.2ms	13.1ms	8.30ms	5.83ms
6	289ms	72.6ms	32.2ms	20.0ms	12.6ms	8.80ms
8	394ms	99.0ms	44.1ms	23.5ms	16.8ms	12.6ms

A good value for α as been determined experimentally as $25\% \leq \alpha \leq 35\%$. Higher threshold would create too many false negatives whereas a lower threshold would create too many false positives.

3.1.3.3 Optimization

Downsampling The major drawback of this method is inherent in all pattern matching methods: the processing time is very long for one pattern, but increases proportionally to the number of patterns to match, including scaled references. This means that we can only process either a few references with a constant scale or use a wider scale with only one reference. Both reference and motion maps obtained from the webcam are downsampled to reduce the processing time. Table 1 presents the impact of downsampling and the amount of scaling applied to the reference.

Filled References The template presented in Figure 8 were sampled manually from a difference frame of a user demonstrating a hand gesture. This approach gives correct results; however it can be greatly improved by modifying the templates. The method used to set the threshold is sensitive to the number of white pixels in a reference. If a reference counts many fingers shown, it will have more edges and therefore the $Dist_{Ref \leftrightarrow Ref}$ will be higher, making it very sensitive to noise compared to the other since its threshold will be significantly higher. Therefore we need to level all threshold values. A simple way to do that is to add gray pixels to references. Besides the inside of the hand contains unstable white zones due to the lack of strong edges in this moving part. A very simple way to discard this information is

to fill the inside of the hand references with gray pixels. Since we use a difference between the reference and the image, any value of the image bitmap will give the same value for the inside of the hand, which suppress the problem of unstable white zones. In addition the thresholds will be closer for different references making them less vulnerable to noise.

SAD Calculation Eq. (3) is used to compute the distance between the frame and the reference. It can be seen as a window of the same size as the reference used to extract a region of interest (ROI) shifted by one pixel until the whole frame is covered. For each position the difference between the ROI and the reference is computed. Computing this directly would incur a long processing time. The computational cost of such operation is detailed in Eq. (5).

$$Cost = (N - n) \times (M - m) \times n \times m = O\left(N.M.n.m\right) \tag{5}$$

Where N and M are the height and width of the frame and n and M the height and width of the reference. To reduce the cost of this operation, Eq. (3) is transformed in Eq. (6).

$$Dist\left(x, y\right) = \sum_{x',y'} \Big(\left(Img\left(x + x', y + y'\right)\right)^2 - 2 \times Img\left(x + x', y + y'\right) \cdot Ref\left(x' + y'\right) + \left(Ref\left(x' + y'\right)\right)^2 \Big) \tag{6}$$

The middle term $2 \times Img\left(x + x', y + y'\right) \times Ref\left(x' + y'\right)$ can be computed using DFT. This term can be written as the inverse DFT of the produce of the DFT of Img and Ref as shown in Eq. (7).

$$Img \times Ref = DFT^{-1}\left(DFT\left(Img\right) * DFT\left(Ref\right)\right) \tag{7}$$

Where DFT is computed the $O\left(N^2 logN\right)$ FFT algorithm. The two other terms Img^2 and Ref^2 are added using images integral which reduces the computation to $O\left(N.M\right)$.

To optimize the processing time, we need to do the least operation per frame and per template in the processing loop. Therefore we compute $FFT(Ref)$ and $\int Ref$ for each

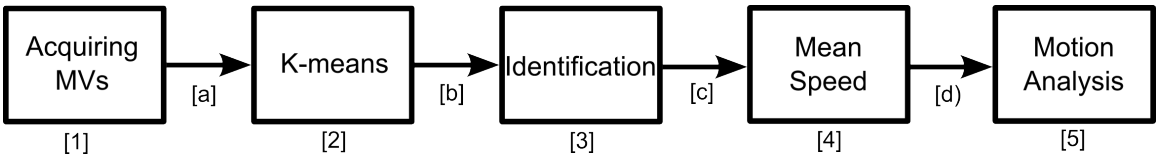


Figure 9: Flowchart of gesture recognition using the H.264 motion vector method.

reference at each downsample rate before starting the recognition. For each frame the $FFT(Img)$ and $\int Img$ are computed once. Then the final difference map is computed for each reference.

3.1.4 H.264 Motion Vector Based

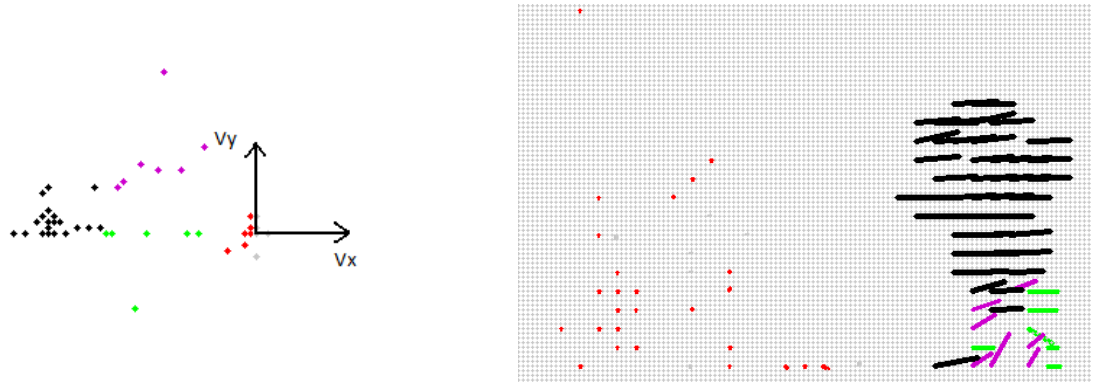
3.1.4.1 Principle

As the gesture and motion we aim to recognize are simple, employ a new method relying on H.264 compressed video's motion vector (MV) to perform real time gesture recognition. This method is called "H.264 Motion Vectors".

H.264 employs motion vectors of macroblocks (MBs) for video compression purposes. Our objective is to employ a H.264 webcam so that we can rely on the hardware device to perform the computing intensive part of the image processing. The rest of the processing can be performed by a broadband-enabled HDTV or a computer.

A related method for object detection using compressed H.264 videos is reported in [18]. The authors in [18] estimate the global motion to discard motion generated by a moving camera. Then, they use Motion History Image to identify and track objects. Object History Images allow them to estimate objects trajectories along the video sequence. This method was designed to be very generic. It can handle several moving objects but no information is extracted to characterize the nature of the object. In this section, we show how this can be done.

We want to specifically detect hand motion (trajectory) with a static webcam. Therefore, we do not need to use global motion estimation but we need to make assumptions on the hand characteristics such as the size and speed of movement. Moreover, we want to distinguish the hand from the rest of the body, which can also move. Figure 9 presents the flowchart of this method.



(a) Example of speed domain clustering (b) corresponding spatial domain clustering

Figure 10: Motion vector clustering using K-Means.

3.1.4.2 Description

In this method, we first extract the motion vectors in the compressed video. Then, we isolate the different moving objects. We cluster the motion vectors by range and direction using the K-Means method. This algorithm aims to partition a data set by minimizing the magnitude of the vector projected from the center of the cluster in Eq. (8).

$$\sum_{i=0}^{N}\sum_{j=0}^{M_i}\left|Pt_j\vec{C}_i\right|^2 \tag{8}$$

Where N is the number of cluster, M_i the number of motion vectors in cluster i, C_i the center of cluster i and Pt_j the j^{th} point from cluster i. We divide the motion vectors into 5 clusters and proceed with 5 iterations. Figure 10 presents the result of K-Means in the speed domain (Figure 10(a)) and the space domain (Figure 10(b)). This gives 5 motion vectors clusters which represent 5 objects moving in different directions at different speeds. We use a median filter on these objects to suppress isolated vectors. As the hand and the arm should move in the same direction, we calculate the spatial dispersion of each cluster of vectors. To identify the hand, we set a threshold for the size and the dispersion of the clusters. The hand should have a minimum size to be detected and big objects or dispersed object can be reasonably rejected as potential hands. Then, if several clusters remain, we choose the one with the highest speed, which is the most likely to be the hand.

Figure 11: Screenshots from the test sequences.

Table 2: Processing time for each method.

Method	Processing Time
Skin Color	27.1 ms
Learning	25.6 ms
Template Matching	36.7 ms
TM (optimized)	13.8 ms
H.264 Motion Vectors	7.0 ms

3.2 Results of Hand Segmentation

We tested our methods using different recorded sequences using different webcams, lighting and background. Figure 11 presents screenshots from test sequences.

3.2.1 Computational Efficiency

3.2.1.1 Processing Time

The processing time of each method is presented Table 2. This processing time accounts for the detection of two gestures (palm and finger) with a scaling range of 3:1. Since the learning method only detects the palm gesture, we multiplied its processing time by 2.

The H.264 method presents the best processing time. This is because the computationally intensive step is the computation of the motion vectors which is done before our method is applied. From this results, we can note that all methods can run in real time with two

Table 3: Memory use for each method.

Method	Memory use
Skin Color	0.97 Mo
Learning	4.83 Mo
Template Matching	4.21 Mo
TM (optimized)	4.21 Mo
H.264 Motion Vectors	6.72 Mo

Table 4: Recognition and error rates for each method.

Method	Correct positives	False positives
Skin Color	81%	1.5%
Learning	89%	0.17%
Template Matching	23%	0.83%
TM (optimized)	95%	0.70%
H.264 Motion Vectors	67.5%	5.4%

gestures. The impact of optimization on the template matching method is to decrease the processing time by almost a factor of 3.

3.2.1.2 Memory Use

We present in Table 3 the memory use for input and output of the methods.

We observe that the skin color method has a very low memory load because it uses a single channel image and a one dimension histogram.

3.2.2 Recognition and Error Rates

For each sequence, we compute the number of detection and false positives. Since the learning method is only trained to recognize the palm gesture, we did not use finger sequences to test it. Table 4 presents the percentage of successful recognitions and and the percentage of false positives for all test sequences.

The skin color algorithm is given a sample of skin color at the beginning of each sequence using the Template Matching method to get the palm position and extract the skin

Table 5: Pixel precision for each method.

Method	Precision
Skin Color	16 pixels
Learning	2.5 pixels
Template Matching	12 pixels
TM (optimized)	12 pixels
H.264 Motion Vectors	23 pixels

color histogram. The low number of correct positives can be explained by an inaccurate sampling of the skin, resulting in a poor performance of the algorithm. For this method, the false positives are generated by the detection of the head instead of the hand. The learning method is sensitive to background with the same grayscale value as the hand because it processes grayscale images. Since the test sequences did not contain a lot of these backgrounds, the performances are very good. This method achieves a very good rate of false positives, the only false positive reported are due to the detection of the same hand at different scales. The template matching method benefit from the optimization and achieves good performances. The H.264 method does not perform as well as other methods because it cannot rely on any template. In addition it sometimes assimilates moving objects to a hand.

3.2.3 Precision

The precision of the position detection is evaluated by comparing the maximum variations of the position of the detected hand compared to a manually set absolute reference. The results are presented in Table 5.

We can observe that the H.264 lacks precision due to the size of the macroblocks. The skin color achieves good performances with a covered arm. However, when the skin of the arm is visible, the maximum ratio set between the width and the height of the hand tends to be biased, causing the lack of precision. The template matching method accuracy depends on the position of the hand. Because we use different scaled templates to match the hand, being at the limit between two scales will result in poorly precise hand location. On the

other hand, the Haar cascade method always provides an accurate detection because it averages the results of different scale detection.

3.3 Trajectory Analysis

The main purpose of our hand gesture recognition is to create a human TV interface. To control a TV, different paradigms are possible. The cursor metaphor which is usually associated with a computer WIMP interface will be developed in Subsection 5.2.1. This interface does not require any trajectory analysis. However we want to provide a more natural way to interact using moving hands. To do this, trajectory analysis is needed and this must be independent of the range (i.e., the result should be the same regardless of the distance between the user and the webcam). Indeed, if the user is far away, the gesture trajectory will appear smaller but should yield the same result.

3.3.1 Histogram Matching Letter Recognition

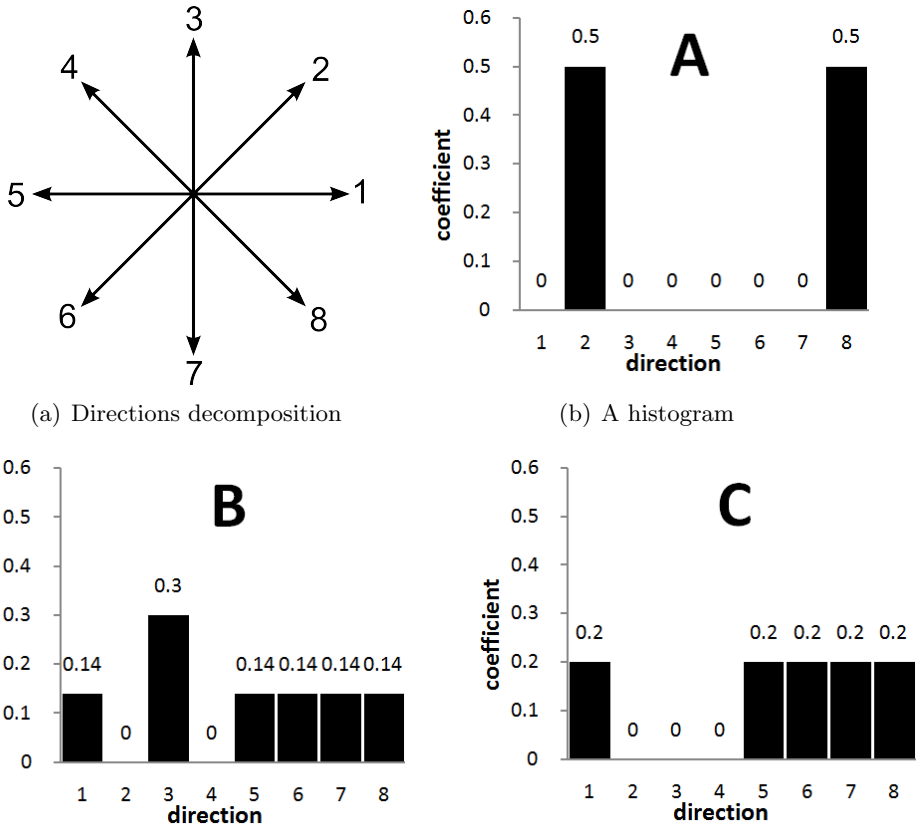
3.3.1.1 Principle

We now test our trajectory recognition algorithm using the histogram matching method. In [15], time segmentation is done by using tracking the hand gesture at the start of the motion and at the end. However, because we are now using H.264 motion vectors, segmentation is performed manually as follows. We first compute the speed and allocate the value to one of the 8 directions presented in Figure 12(a). We store this value in a histogram. We then compare our histogram to the reference histograms displayed in Figure 12(b), Figure 12(c) and Figure 12(d).

3.3.1.2 Results

Table 6 shows the results for the gesture recognition of 5 trajectories, with some trajectories representing letters A, B, and C. We observe that the Motion method is not very efficient because it misses most of the vertical motion. The Color method and the H.264 motion vectors method present goods results. The excellent results of H.264 method can be attributed to the accuracy of the speed calculation as described in the previous paragraph.

This method is efficient for few trajectory directions. However trying to recognize every



(a) Directions decomposition

(b) A histogram

(c) B histogram

(d) C histogram

Figure 12: Possible directions of vectors and histogram examples for A, B and C.

Table 6: Trajectory recognition rate using the histogram matching method.

	gesture recognition rates		
	Motion	Color	MV
right	100%	100%	100%
Up	0%	100%	100%
A	27%	100%	100%
B	78%	78%	100%
C	38%	92%	100%

letter in the alphabet would lead to confusions. For example the letter 'D' and the letter 'O' would have very similar histograms.

3.3.2 Fourier Descriptors Letter Recognition

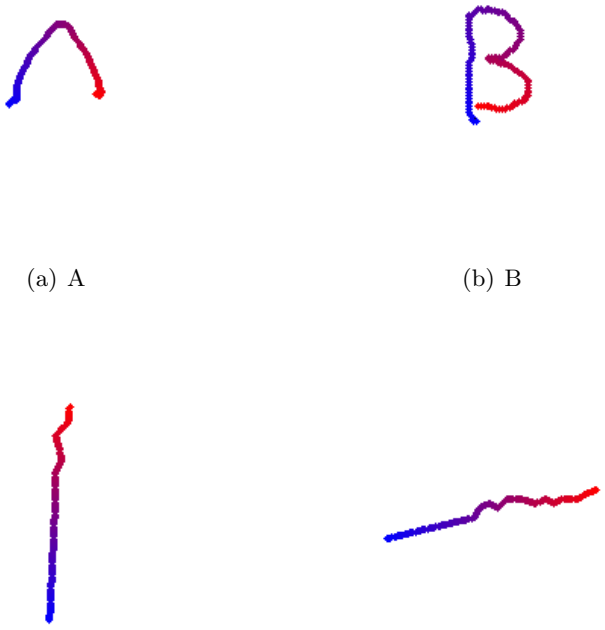
3.3.2.1 Principle

To overcome the problem of the method described in Subsection 3.3.1, we use the Fourier Descriptors presented in Subsection 3.1.1 to describe a contour. Fourier Descriptors have the advantage of being scale invariant and translation invariant, which is suitable for our trajectory analysis, since the gesture can be scaled in different parts of the frame. However we cannot directly use Fourier Descriptors for the contour because gestures trajectories are not necessarily cyclic. We need to do a temporal segmentation first to know when the gesture starts and when the gesture ends. We combine three methods to do this segmentation. The first is based on the hand shape. If the user changes its hand gesture, we assume that the trajectory is terminated and start to analyze it. For ease of use, we provide two gestures to the user, one to perform the trajectory, and one to end and validate it. The second way to determine if the trajectory is terminated is to monitor the trajectory length. If it exceeds a certain threshold, we assume that the desired gesture is completed. The last temporal segmentation method relies on a timer which is reset every time the hand is detected. If no hand is detected during a sufficient amount of time, we conclude that the user has completed his gesture. A good value for this amount of time has been determined experimentally as $300ms \leq t \leq 400ms$. Figure 13 presents example of trajectories successfully identified. The trajectory is colored for a better visualization. It starts with the blue dots and gradually change color to end with the red dots.

3.3.3 Scrolling Method

3.3.3.1 Principle

We propose a scrolling method which can be applied to the browsing interface similar to the one described in Subsection 5.2.2. The idea is to control an interface using 2D movements, either horizontal or vertical. The purpose of this interaction method is to get a very responsive interface. Motions are divided in two directions: horizontal and



(a) A (b) B

(c) Up (d) Right

Figure 13: Trajectory analysis of different patterns using Fourier Descriptors.

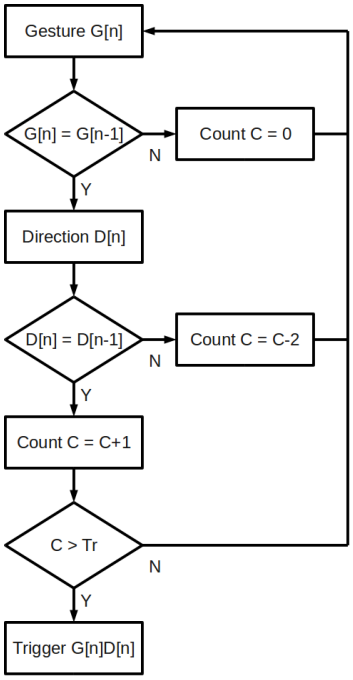


Figure 14: Flowchart of the scrolling method counting.

vertical. When a gesture associated to a direction is detected, we trigger a message to control an interface. This gives four directions multiplied by the number of gesture command possibilities. To analyze the direction, we compute the difference between two consecutive detected hands, which is robust to false negative detection. If no hand is detected, the algorithm is not activated. Since the movement detection is based on the pixel location of the hand, the direction value may be noisy. To overcome this, we use additive increase and weighted additive decrease. We increment the gesture counter by one for the correct direction and for a wrong direction, we decrease it by two. Therefore a single wrong direction created by the noise will have a limited impact and a wrong gesture direction have very few chances to be detected. Figure 14 presents the flowchart of the method adopted to achieve a very responsive interface.

This method presents the advantage to be scale independent and translation invariant without any additional computing. Indeed we only use the relative positions of the hand location, which makes the method independent of the starting location. We also increment independently from the the dist

Table 7: Delay to recognize a left gesture for each trajectory method.

Method	Delay
Histogram	0.4 s
Letter (two gestures)	0.4 s
Letter (single gesture)	0.7 s
Scrolling	0.1 s

3.3.3.2 Results

Since the goal of this method is to obtain the most responsive interface, we tested the delay between using the template matching hand detection method. The delay to recognize a left horizontal gesture is compared using the three methods. Table 7 presents the results for each method.

The letter trajectory can be ended either by performing another gesture as for the histogram method, of by waiting for a timeout using a single gesture. We can observe that using two gesture allows to speed up the trajectory recognition for the two complex trajectory recognition methods. However, the scrolling method allows a very quick recognition because it does not require the trajectory to end before it can be processed. Therefore this last method provides a very responsive manipulation.

CHAPTER IV

STEREO WEBCAM HAND GESTURE RECOGNITION

4.1 Previous Work

Hand gesture recognition can benefit from the additional knowledge of the depth component, which can be obtained via a stereoscopic webcam. As a single webcam only captures a 2D image, further processing needs to be done to evaluate the depth of the image. Methods using a single webcam to evaluate a 3D image have been reported. Top-down pointing cameras can be used to limit the amplitude of the user movements but it requires a simple background for segmentation. Dhawale et al. [5] propose to use skin color segmentation followed by a width and height estimation. The hand posture and motion is then estimated from these distances. Segen et al. [29] use a local feature classifier to control a simple interface with a very low error rate. Although these approaches offer good results for simple hand motion, tey rely on hand segmentation that cannot be applied against other types of complex background (e.g., in a living room) and with other camera positions such as front facing cameras.

Stereo webcams that process two image streams are now available and will assume more importance with 3D Skype video calling. A stereo 3D webcam consists of two webcams placed side by side, separated by approximately the distance between two eyes. Sangi et al. [28] use a rectified stereo webcam to estimate the depth of markers attached to the hand. Their segmentation is based on skin color detection using a histogram representation and the hand needs to be the only object in front of the stereo camera. Methods using both face or head detection have been reported [10, 23]. Skin color is also used for segmentation and a depth map is created to estimate he direction where the hand is pointing. However, using the head limits the capabilities of the gesture detection since the users head is not always visible from to the camera. Another approach is to use the depth information to segment the hand. Hongmo et al. [15] use skin colored region coupled with depth information to extract

the hand by assuming that the centroid of the closest region that contains skin color as the hand position. They discard the head region by making an extra step of head detection and removing a large region around the head. However this method is very sensitive to false detection. Since it relies on head detection, if the head is not correctly detected, the hand detection will fail. In addition, only one hand should appear for this method to work and the result of this segmentation may be prone to noise.

4.2 Our Methods

4.2.1 Depth Map

The OpenCV library provides a framework to calibrate single and stereo camera [2]. However, we used the Minoru 3D webcam which is available commercially at low cost and does not require the extra calibration step. Nevertheless, we tested the accuracy of the calibration. By calibrating each webcam view using a chessboard, the high quality of the lenses do not create any radial or tangential distortion. The stereo calibration using several pairs of images of the chessboard also lead to the conclusion that the webcam does not require these extra steps. The Hartley's algorithm [12] allows us to conclude that the webcam are correctly aligned and therefore does not require calibration.

A stereo webcam does not provide a 3D model of the scene but rather a 2D view with a depth information. This depth information is computed by observing the correlation between the two frames extracted from each webcam. Similarities are found in the two frames as part of a similar object. From the difference of viewing angle of the same object between the left and the right frame, the depth of the object can be estimated. From this depth map, we can estimate the relative distance of different objects from the camera. A matching algorithm to use may include key points matching and macroblock matching. We choose to use the macroblock matching method because we wish to extract the area of the object. Hirschmuller [13] has developed a very robust algorithm for stereo block matching which results in a high-precision depth map. However, this algorithm cannot run in real time on a computer. Hence, we chose the more simple block matching algorithm described in [2]. The stereo matching algorithm represent the bottleneck of the stereo processing,

using the block matching method, we obtain a frame rate of 16.2 fps which is equivalent to a processing time of 61.7ms per frame.

4.2.2 Simple Sampling Method

We propose a simple application for the stereo webcam, extracting a hand reference to calibrate other methods requiring some training. This method aims at extracting a hand sample from the user prior to any knowledge concerning the hand gesture, the hand posture or the hand color.

4.2.2.1 Description

From the two images generated by the webcam, a stereo matching algorithm is used to extract a depth map of the viewed scene.As we want to extract a hand reference, a simple way to segment it is to have the user reach out his hand in front of the camera. In this configuration, the hand is located in the first plane of the webcam image, the user in the second plane, and the background in the next plane. To segment the hand from this configuration, we employ the Otsus thresholding method [24], which allows the depth map to be divided in two regions: the first plane where the hand is located and the other planes. However, the binary mask obtained from the previous step contains a lot of noise generated by the stereo matching method. To overcome this, we successively erode and dilate the image several times to reduce the noise area relative to the object area representing the hand. To estimate the position of the hand, we use the centroid of the binary mask. To estimate its size, we use a value proportional to the second order moments of the binary mask. From these parameters, the region surrounding the hand can be extracted.

Since the stereo matching algorithm gives a noisy depth map, it cannot be used to segment the hand region. A background model is used instead and this gives a more precise and accurate spatial segmentation. The background model must be established before the user starts to show his hand in front of the camera. To do this, the user stands in front of the webcam without showing his hand for a few seconds, and then brings out his hand to be recorded. The chosen background modeling technique is the Mixture of Gaussians [30] because it trains quickly and gives accurate results for this simple use. Once the background

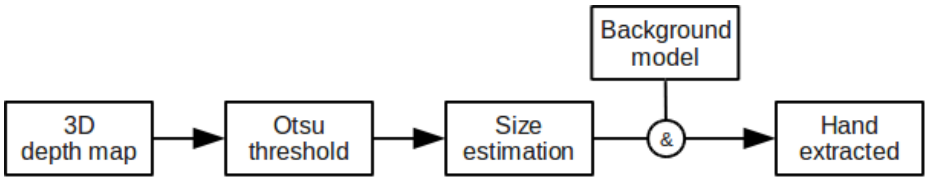
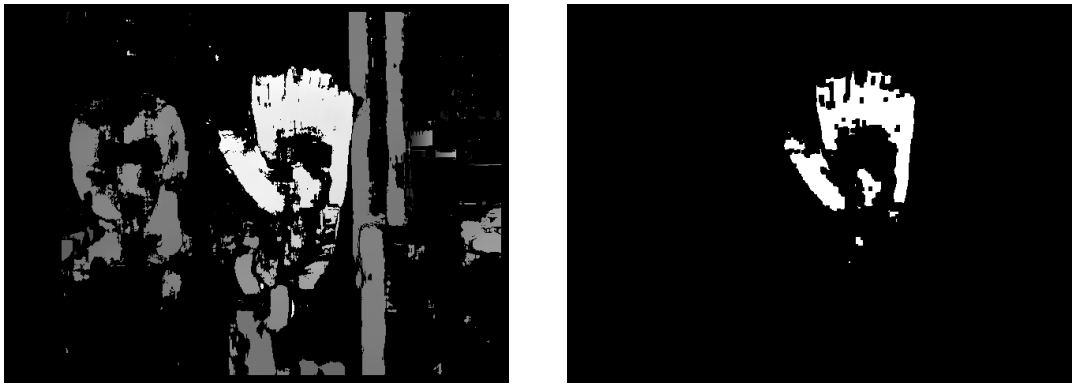


Figure 15: Flowchart of the method.



(a) Depth map

(b) Binary mask after Otsu thresholding and noise filtering

Figure 16: Depth map segmentation.

model is established, the algorithm attempts to identify any object in the first plane. To ensure that the identified object is the hand, we use two criteria. We verify that the object size is larger than the noise but smaller than half of the frame. The second criterion is that the object is static. We can then check the hand position for 30 consecutive frames. The steps to extract the hand are summarized in Figure 15.

4.2.2.2 Results

We have been able to successfully extract hand gestures from various scenes, light environments, and users. Figure 16(a) shows an example of a depth map generated by the stereo matching algorithm. The grayscale areas correspond to the relative depth of the object. The light areas are closest to the webcam whereas the dark areas are further back. The black area represents the zone where no reliable depth estimate is found and thus, no depth information is produced. This usually happens with mono colored walls. Figure 16(b) shows the image after Otsu thresholding and noise filtering are applied. The users hand is correctly segmented between the first plane and the user/background planes.

(a) Hand extracted (b) Backgound model (c) Area selected

Figure 17: Hand extraction.

Figure 17 presents the results of the hand extraction steps. Figure 17(a) shows the region extracted using the centroid and the moments of Figure 16(b). The area selected is correctly fitted to the hand size. Figure 17(b) shows the binary mask generated by the background model on the region. The white pixels represent the foreground and the black pixels represent the background. It can be noted that the image includes some noise at the bottom of the hand but overall the hand is correctly segmented. Figure 17(c) illustrates the superposition of the two previous images (i.e., the extracted and segmented hand).

From the segment hand, several parameters can be extracted such as the color distribution, the hand shape, the hand gesture or the ambient lighting.

The background modeling allows segmenting correctly the hand most of the time. However, if the user appears in the segmented area, it will affect the segmentation accuracy. Indeed the user is not part of the background, therefore if a part of his body, for example his head, is aligned with his hand and the webcam, it will appear as part of the area selected. After segmentation, both the head of the user and the hand will appear in the extracted image and needs to be separated.

Our method offers an original way to automatically detect and segment a hand. It does not rely on any assumption about the hand characteristics (e.g., size, skin color) and can be applied in any configuration. It can be used as a calibration step for other hand gesture recognition methods or to create a hand database for machine learning purposes.

Table 8: Range viewing capabilities using Otsu thresholding method.

$H \leftrightarrow U$ \ $U \leftrightarrow W$	0.5m	1.0m	1.5m	2.0m	2.5m	3.0m
10cm	✗	✗	✗	✗	✗	✗
20cm	✗	✗	✗	✗	✗	✗
30cm	✗	✗	✗	✗	✗	✗
40cm	✓	✗	✗	✗	✗	✗
50cm	✓	✓	✓	✗	✗	✗

4.2.3 Window Thresholding Method

One of the limiting factor of the previous method is the thresholding method, The user has to reach out his hand in front of him and stand close to the webcam to enable the recognition. Table 8 presents the results of hand segmentation for different hand positions using the Otsu thresholding method.

4.2.3.1 Principle

To overcome this limitation, we have developed an original approach to obtain the threshold of the depth map. We first apply a very simple color mask to eliminate part of the background. We then decompose the depth map into an histogram. The x-coordinate of the histogram represents the value of the depth map and the y-coordinate represents the number of pixels at the depth. The hand should appear as an isolated peak close to the webcam. To detect this peak, we shift a window along the depth values, starting from the closest point to the webcam. For each value of the depth, we compute the sum of the neighboring values of the histogram. We continue to slide the window until we reach a value approximately equal to the area of the hand for this depth. It should be noted that the depth map is computed as a disparity map. A disparity map represents the difference between the positions of an object on the left and right frame in pixels. The closer the object to the webcam, the higher the disparity. This implies that the precision is reduced and the noise is more important for far distances. The hand width and height are proportional to the value of the disparity map at the hand location. The computation of the hand is

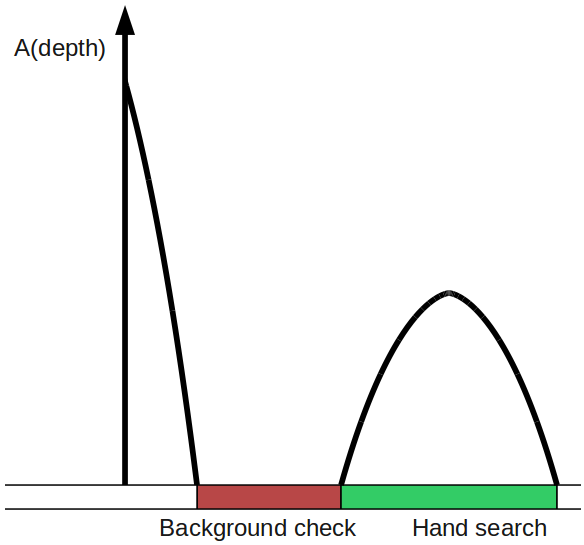


Figure 18: Scheme of the window principle.

therefore proportional to the depth where the window is located. If we reach the hand size, we need to check whether we have detected the hand or the beginning of the background. We use a second window starting just after our first window to compute the number of direct neighbors deeper than the potential hand. If the sum is greater than the hand area, this means that we have detected an object too large to be the hand, which we classify as background. In that case we shift the window until we reach the end of the background and restart t o search for the hand. If no background is detected, we have found the depth of the hand. Figure 18 presents the window and the value of the histogram if the hand is present. The $A(depth)$ stands for area of this depth since the value of the histogram represents the area of the pixels having this depth value.

The algorithm can be summed up by the following pseudo-code:

```
while depth < depthMax
   sum = integral(windowHand(depth) & histogram)
   if(sum ~ handArea)
      sum2 = integral(windowBackground(depth) & histogram)
      if(sum + sum2 ~ handArea)
         return depth
      else
```

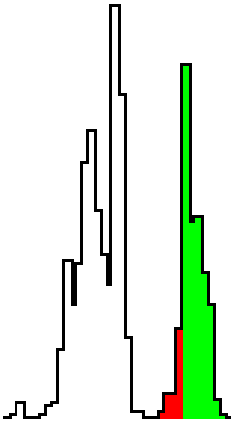


Figure 19: Histogram showing the successful segmentation.

```
reduce depth until integral(windowHand(depth) & histogram) << handArea
```

Since the depth map is actually a disparity map, if we want the window to cover the same thickness for any depth, we have to scale the size of the window. The size of the window is given by Eq. (9).

$$windowSize = \alpha \frac{depth}{1 + \beta depth} - \gamma depth \tag{9}$$

Where α, β and γ are parameters which depend on the webcam and are determined experimentally.

4.2.3.2 Results

Figure 19 shows an example of successful segmentation using our window method.

Using this method yields the capability of extracting an object (which is not the hand) in the foreground and enabling a larger range of detections. However, the range is still limited because the stereo webcam loses precision as the depth increases. Table 9 presents the results of hand segmentation for different position of the hand and the user using the window method.

Table 9: Range viewing capabilities using window method.

$H \leftrightarrow U$ \ $U \leftrightarrow W$	0.5m	1.0m	1.5m	2.0m	2.5m	3.0m
10cm	✗	✗	✗	✗	✗	✗
20cm	✓	✗	✗	✗	✗	✗
30cm	✓	✓	✓	✗	✗	✗
40cm	✓	✓	✓	✓	✗	✗
50cm	✓	✓	✓	✓	✓	✗

4.3 Rotation Estimation

The rotation estimation problem is a well known problem in robotics [6, 22]. It usually consists of estimating the camera position from the rotated image. Here, we would like to only estimate the rotation of the hand in front of a static background. To avoid the constraints of background modeling, we propose a new approach that does not rely on the distinction of the background from the foreground. This method relies on 3D analysis.

The 3D analysis gives a region of interest ROI containing the hand from which we extract the rotated hand. We propose this method to estimate the rotation angle of the hand.

We apply a first derivative order Sobel filter to the ROI to obtain the edges of the ROI in each direction as showed in Eq. (10) and Eq. (11):

$$S_x(x,y) = \frac{\partial ROI(x,y)}{\partial x} = ROI\,(y, x+1) - ROI\,(y, x-1) \tag{10}$$

$$S_y(x,y) = \frac{\partial ROI(x,y)}{\partial y} = ROI\,(y+1, x) - ROI\,(y-1, x) \tag{11}$$

Where $0 \leq x \leq ROI_width$ and $0 \leq y \leq ROI_height$. Applying this simple filter would create noise resulting in an excessive number of angles in multiples of 90∘. The Sobel kernel used is 7×7 pixels weighted by a Gaussian smoothing mask.

From the result of the Sobel filtering, we compute the angle A and the amplitude K for each pixel in Eq. (12) and Eq. (13).

$$A(x,y) = arctan\left(\frac{S_y(x,y)}{S_x(x,y)}\right) \tag{12}$$

$$K(x,y) = \sqrt{S_x(x,y)^2 + S_y(x,y)^2} \tag{13}$$

Using the previous vectors, we can compute the probability distribution function p for each degree angle a, $a \in [|0, 359|]$ in Eq. (14).

$$p(a) = \frac{\#\{x, y \| A(x, y) = a\}}{H_{ROI} \times W_{ROI}} \tag{14}$$

Where H_{ROI} and W_{ROI} are respectively the height and the width of the ROI and # stands for cardinal.

The weighted probability distribution function would be P in Eq. (15).

$$P(a) = \frac{\sum_{\{x,y \| A(x,y)=a\}} K(x, y)}{\sum_{x,y} K(x, y)} \tag{15}$$

However, dividing by $\sum_{x,y} K(x, y)$ to have the probability would prevent us from distinguishing noise from a strong edge signal. Therefore, we choose to use the signal s defined in Eq. (16).

$$s(a) = \sum_{\{x,y \| A(x,y)=a\}} K(x, y) \tag{16}$$

This signal can be written as the sum of 3 signals as shown in Eq. (17).

$$s(a) = b(a) + h(a) + noise(a) \tag{17}$$

Where b is the signal containing the edges from the background which are static, h contains the edges of the hand which are rotating, and noise is the combination of all types of noise resulting from the numerical approximations and the variations of b and h.

To compute the speed, we need to compare successive frames. If t is the current time and Δt the time between two frames, we define n in Eq. (18).

$$n = \frac{t}{\Delta t} \tag{18}$$

For a frame rate of 25fps, $\Delta t = 40ms$.

Our signal at the frame n is shown in Eq. (19).

$$s_n(a) = b_n(a) + h_n(a) + noise_n(a) \tag{19}$$

We consider the noise negligible compared to the other signal. The hand is rotating by γ between the frames $n-2$ and $n-1$, and by θ between the frames $n-1$ and n. We then

obtain Eq. (20), Eq. (21) and Eq. (22).

$$s_n(a) = b_n(a) + h_n(a) \tag{20}$$

$$s_{n-1}(a) = b_{n-1}(a) + h_{n-1}(a - \theta) \tag{21}$$

$$s_{n-2}(a) = b_{n-2}(a) + h_{n-2}(a - \gamma) \tag{22}$$

Note that to keep the notation simple, we use circular signals, whose periodicity is 360° as shown in Eq. (23), Eq. (24) and Eq. (25).

$$s(\alpha + 360^\circ) = s(\alpha) \tag{23}$$

$$b(\alpha + 360^\circ) = b(\alpha) \tag{24}$$

$$h(\alpha + 360^\circ) = h(\alpha) \tag{25}$$

We want to estimate θ, the angle of rotation of the hand between the last two frames. The previous equations contain more unknown variables than the number of equations we have. Therefore we have to make some assumptions about the angle distribution. The first assumption (Eq. (26) and Eq. (27)), as we mentioned before, is to consider b and h constant over 3 successive frame. This is justified because the background do not change from one frame to the other except the small parts occluded by the hand and the hand shape does not change dramatically while executing a rotation. The second assumption (Eq. (28)) is to consider that the speed of the hand is constant between three consecutive frames. This is not true during the acceleration and deceleration phases at the beginning and the end of the gesture, but is acceptable during the middle of the gesture given that the time between 3 frames at a frame rate of 25 fps is $2 \times \Delta t \approx 80ms$.

$$b_n \approx b_{n-1} \approx b_{n-2} \tag{26}$$

$$h_n \approx h_{n-1} \approx h_{n-2} \tag{27}$$

$$\gamma \approx 2 \times \theta \tag{28}$$

If we insert these approximations in our equations we get Eq. (29), Eq. (30) and Eq. (31).

$$s_n(a) = b(a) + h(a) \tag{29}$$

$$s_{n-1}(a) = b(a) + h(a - \theta) \tag{30}$$

$$s_{n-2}(a) = b(a) + h(a - 2\theta) \tag{31}$$

To suppress the term b describing the background, we need to subtract signals with each other as shown in Eq. (32). We also introduce here a shifting angle φ in Eq. (33).

$$s_n(a) - s_{n-1}(a) = h(a) - h(a - \theta) \tag{32}$$

$$s_{n-1}(a + \varphi) - s_{n-2}(a + \varphi) = h(a + \varphi - \theta) - h(a + \varphi - 2\theta) \tag{33}$$

In particular for $\varphi = \theta$ we obtain Eq. (34) and Eq. (35).

$$s_n(a) - s_{n-1}(a) = h(a) - h(a - \theta) \tag{34}$$

$$s_{n-1}(a + \theta) - s_{n-2}(a + \theta) = h(a) - h(a - \theta) \tag{35}$$

We define r as the inter correlation between the two previous signal as shown in Eq. (36).

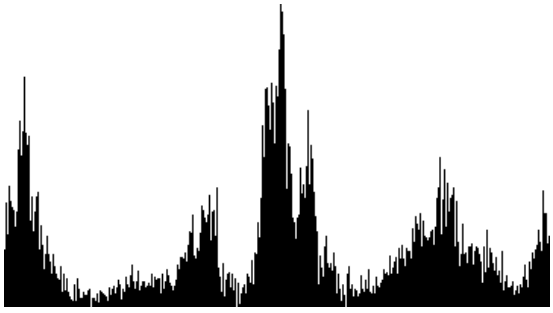
$$r(\varphi) = \sum_{a \in [|0,359|]} (s_n(a) - s_{n-1}(a)) \times (s_{n-1}(a + \varphi) - s_{n-2}(a + \varphi)) \tag{36}$$

Since the two signal are identical for $\varphi = \theta$, we can resolve θ as being the angle φ for which the correlation is maximum. Eq. (37) gives the solution to our problem.

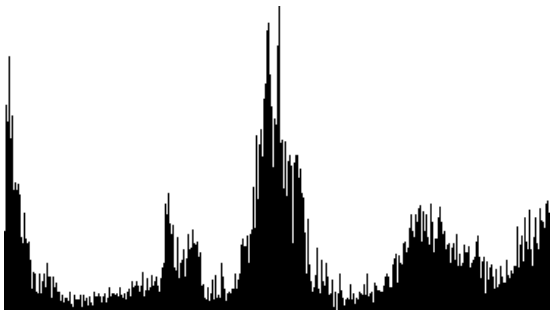
$$\theta = \underset{\varphi \in [|0,359|]}{\arg\max}\, r(\varphi) \tag{37}$$

Figure 20 presents the result of this method. We can easily observe from Figure 20(c), Figure 20(b) and Figure 20(a) that the constant signal b around 180 (middle of the histogram) and the signal h shifting from right to left particularly are visible around 0. The resulting r in Figure 20(d) displays a peak in $\theta \approx 5°$ which is the wanted value.

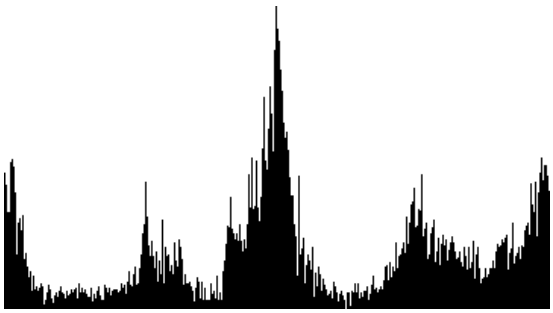
The estimated rotation is not very accurate in predicting the exact angular speed of the hand, but gives a good approximation of the angle and the correct value of the rotation direction.



(a) s at time n-2



(b) s at time n-1



(c) s at time n

(d) r

Figure 20: Rotation estimation results.

4.4 Hand pointer

4.4.1 Principle

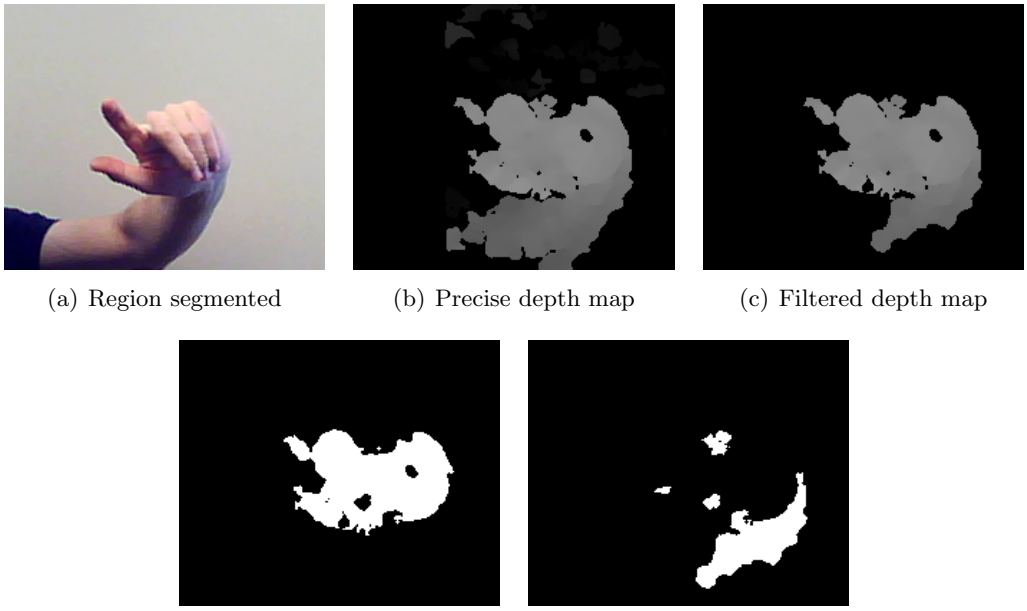
We propose a method to recognize the hand pointing gesture using the 3D webcam data. Using the hand segmentation method described in Subsection 4.2.3, we extract the region of the frame containing the hand. This region is presented in Figure 21(a). Since this region is less than 100×100 pixels, we can apply the Hirschmuller [13] to get a more precise depth map. This map is displayed in Figure 21(b). It is then processed to extract the hand pose. First we extract the depth histogram. To remove the noise, we discard every pixel which depth is under $\mu - \sigma$ or over $\mu + \sigma$ where μ represents the mean value of the depth map and σ represents the standard deviation of the depth. This process can be applied several times if the depth map contains a lot of noise. The result of the filtering is shown in Figure 21(c). We then divide the filtered depth map in two binary maps. The first is such that $depth \leq \mu$ represents the region close to the webcam as shown in Figure 21(d). The second is such that $depth \geq \mu$ represents the region away from the webcam as shown in Figure 21(e). We then compute the centroid for each binary map and the average depth for each of these maps. The direction of the pointing hand can be estimated by $\vec{V}$ defined in Eq. (38).

$$\vec{V} = \begin{bmatrix} C_{1x} - C_{2x} \\ C_{1y} - C_{2y} \\ depth_1 - depth_2 \end{bmatrix} \tag{38}$$

Where C_1 designate the centroid from the first binary map and C_2 designate the centroid from the second binary map.

4.4.2 Results

As it can be seen in Figure 21, the algorithm is able to distinguish the arm from the hand and therefore gives correct results (around 70%) when the arm is visible. However the depth map is not precise enough to distinguish the fingertip from the hand. Therefore the algorithm cannot detect a pointing gesture done with the hand only.



(a) Region segmented (b) Precise depth map (c) Filtered depth map

(d) Region close to the webcam (e) Region away from the webcam

Figure 21: Hand pointer pose estimation.

CHAPTER V

HUMAN COMPUTER INTERFACE

5.1 Features

5.1.1 Pan Tilt Zoom

Several webcam drivers allow a more precise control of the webcam picture through the control of the zoom, pan, and tilt parameters. The zoom feature focuses on a certain part of the image. This can be done using either optical or numerical zoom. Panning indicates the horizontal rotation of the webcam and tilting indicates the vertical rotation of the webcam. By extension, when a digital zoom is used, panning and tilting can be used to control the location of the cropped image. Although the OpenCV library provides direct access to the webcam video frame, the interface with the pan, tilt, and zoom is not available. In particular, we used Logitech camera which require additional software drivers to control these features. We used the Logitech Webcam Pro 9000 which has digital zoom capability as well as the QuickCam Orbit AF which has mechanical pan and tilt capabilities.

We provide a framework for this features integrated with our main framework. The idea is to add a feature in the detection where a very large range of scale is allowed to detect the hand. During this feature, we try to recognize a specific gesture to determine the hand size and location. Once a hand is detected, the correct zoom is computed from the hand size detected as well as the correct pan and tilt values to center the hand after zooming. Once the zoom, pan and tilt have been applied, we return to the original gesture detection feature. A timeout is set so that if no gesture is detected during an extended period of time, the algorithm returns to the previous phase, allowing the user to change location or a different user to interact with the system.

5.1.2 Two hands gestures

We discussed a simple way to interact using gesture with our S̈crolling methodïn Subsection 3.3.3. The drawback of this method is that it limits the number of possible gestures. To

increase the numbers of possible gesture, we propose to use two hands at the same time. The detection part stays similar to one hand gesture except that we search for two hands instead of one. The templates used only need to be mirrored to match both hands. As we want to keep our interactions as simple as possible, we allow the user to either perform gestures with his right hand, with his left hand or with both hands at the same time. Single handed gestures are associated to the most common task as it requires less physical movement of the user. Two handed gestures for less common task such as activating options.

For the trajectory analysis, we still use the very simple scrolling method but we adapt for two hands gestures. If a single gesture is performed, the previously described algorithm is used. If the two-hand gesture is preformed, we detect the shape of each and the direction of each hand, multiplying the possibilities.

5.2 Display

5.2.1 Mouse cursor

Most computers are using the Windows Icon Menu Pointer paradigm as interface, therefore people are used to manipulate a mouse cursor to interact with a machine. We can easily make use of that by converting our hand detection output to a mouse cursor controller. In our implementation, we control the cursor position according the hand motion. We do not directly convert the hand location to a cursor position because this would imply that the user needs reach the four angles of the frame with his hand to me able to move the mouse over the entire screen, but we rather match the speed of the hand to the speed of the mouse. A gesture is assigned to move the mouse and another is assigned to click and drag. A third gesture can be assigned to control a mouse wheel function.

This method presents the advantage of enabling the user to directly browse the internet using a classic browser. However the precision required to click on a web link is very superior to the precision given by a 640x480 webcam capture. Our experiments showed that this lack of precision impaired the direct use of gesture as a mouse.

Figure 22: Video browsing interface.

5.2.2 Video browsing interface

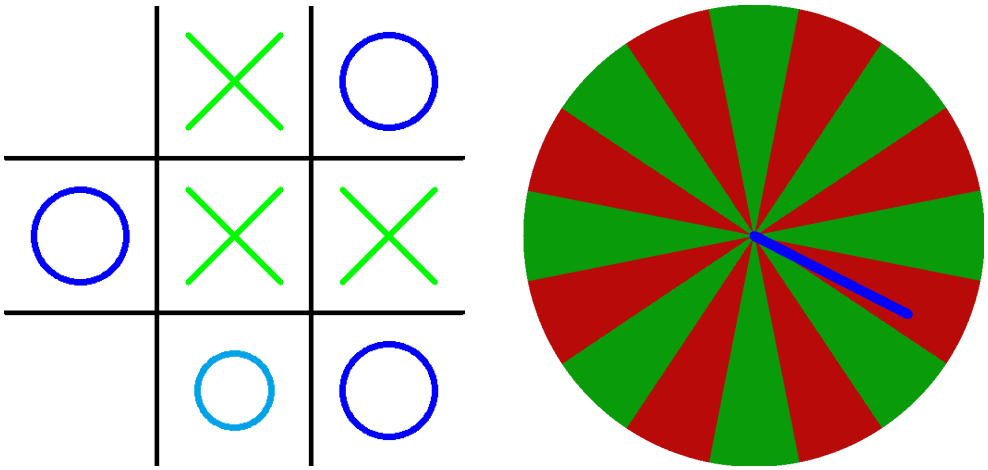
Our main goal is to create a video browsing interface for online video. The scrolling method described in Subsection 3.3.3 allows an efficient control using both gesture and direction. We therefore designed a video browsing widget controlled by this scrolling method. Lateral movements using the finger gesture allow the user to select different videos. A visual feedback is given by enlarging the currently selected video. The palm gesture is used to select and play a video (right motion) or display a caption (left motion). Figure 22 presents a screenshot of the browser window.

5.2.3 Gaming interface

Our hand gesture can easily be applied to different domain. As a proof of concept, we give here two examples of mini-games controlled by our system.

5.2.3.1 Tic Tac Toe

This game is the well-known tic tac toe game. The user is using the circle and playing against the computer which controls the cross. To select a move, the user slides a little circle using the finger gesture and the four scrolling directions. To validate a move, the user simply shows his palm. The game user interface window is demonstrated in Figure 23(a)



(a) Hand extracted (b) Roulette

Figure 23: Game interfaces.

5.2.3.2 Roulette

To make use of the rotation information computed in Section 4.3, we have created an interface dedicated to rotation visualization. The user's hand rotation causes the rotation of the blue line around the roulette wheel. The stating rotation speed is proportional to the hand rotation speed and it slows down until it stops on a random color. The line rotation direction is the same as the hand rotation direction. Figure 23(b) presents a screen capture of this interface.

CHAPTER VI

CONCLUSION

We have applied existing methods including color segmentation and Haar cascade learning to hand segmentation and we have created new methods including motion analysis and depth map thresholding. We have proposed a comparative evaluation of these methods in terms of precision and computational cost. We have analyzed different methods for trajectory analysis and we have presented original human computer interfaces relying on hand gesture recognition.

Future research will focus on using a Graphical Processing Unit instead of a Central Processing Unit to achieve better frame rates.

REFERENCES

[1] Alon, J., Athitsos, V., Yuan, Q., and Sclaroff, S., "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, pp. 1685 –1699, Sept. 2009.

[2] Bradski, G. and Kaehler, A., *Learning OpenCV*. O'Reilly Media Inc., 2008.

[3] Chai, X., Fang, Y., and Wang, K., "Robust hand gesture analysis and application in gallery browsing," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pp. 938 –941, June 2009.

[4] Chen, F.-S., Fu, C.-M., and Huang, C.-L., "Hand gesture recognition using a real-time tracking method and hidden markov models," *Image and Vision Computing*, vol. 21, no. 8, pp. 745 – 758, 2003.

[5] Dhawale, P., Masoodian, M., and Rogers, B., "Bare-hand 3d gesture input to interactive systems," in *CHINZ '06: Proceedings of the 7th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction*, (New York, NY, USA), pp. 25–32, ACM, 2006.

[6] Falcon, L. and Bayro-Corrochano, E., "Radon transform and harmonical analysis using lines for 3d rotation estimation without correpondences from omnidirectional vision," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1 –6, 2007.

[7] Freeman, W. and Weissman, C., "Television control by hand gestures," in *Proc. Int'l Workshop on Automatic Face and Gesture Recognition*, pp. 179–183, June 1995.

[8] Ghobadi, S. E., Loepprich, O. E., Ahmadov, F., Bernshausen, J., Hartmann, K., and Loffeld, O., *Advances in Visual Computing*, vol. 5359/2008, ch. Real Time Hand Based Robot Control Using 2D/3D Images, pp. 307–316. Springer Berlin / Heidelberg, 2008.

[9] Grimson, W., Stauffer, C., Romano, R., and Lee, L., "Using adaptive tracking to classify and monitor activities in a site," in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pp. 22 –29, 1998.

[10] Guan, Y. and Zheng, M., "Real-time 3d pointing gesture recognition for natural hci," in *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pp. 2433 –2436, 2008.

[11] Hamada, Y., Shimada, N., and Shirai, Y., "Hand shape estimation under complex backgrounds for sign language recognition," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pp. 589 – 594, 2004.

[12] Hartley, R. I., "Camera calibration using line correspondences," in *In Proc. DARPA Image Understanding Workshop*, pp. 361–366, 1993.

[13] Hirschmuller, H., "Stereo processing by semiglobal matching and mutual information," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, pp. 328 –341, Feb. 2008.

[14] Huang, Y., Huang, T., and Niemann, H., "Two-handed gesture tracking incorporating template warping with static segmentation," in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pp. 275 –280, 2002.

[15] Je, H., Kim, J., and Kim, D., "Hand gesture recognition to understand musical conducting action," in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pp. 163 –168, 2007.

[16] Just, A. and Marcel, S., "A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition," *Computer Vision and Image Understanding*, vol. 113, no. 4, pp. 532 – 543, 2009.

[17] Kakumanu, P., Makrogiannis, S., and Bourbakis, N., "A survey of skin-color modeling and detection methods," *Pattern Recogn.*, vol. 40, no. 3, pp. 1106–1122, 2007.

[18] Käs, C. and Nicolas, H., "An approach to trajectory estimation of moving objects in the h.264 compressed domain," in *PSIVT '09: Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, (Berlin, Heidelberg), pp. 318–329, Springer-Verlag, 2008.

[19] Lidoris, G., Rohrmuller, F., Wollherr, D., and Buss, M., "The autonomous city explorer (ace) project x2014; mobile robot navigation in highly populated urban environments," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 1416 –1422, 2009.

[20] Lin, E., Cassidy, A., Hook, D., Baliga, A., and Chen, T., "Hand tracking using spatial gesture modeling and visual feedback for a virtual dj system," in *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, pp. 197 – 202, 2002.

[21] Mitome, A. and Ishii, R., "A comparison of hand shape recognition algorithms," in *Industrial Electronics Society, 2003. IECON '03. The 29th Annual Conference of the IEEE*, vol. 3, pp. 2261 – 2265 Vol.3, 2003.

[22] Moriya, T. and Takeda, H., "Solving the rotation-estimation problem by using the perspective three-point algorithm," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, pp. 766 –773 vol.1, 2000.

[23] Nickel, K. and Stiefelhagen, R., "Pointing gesture recognition based on 3d-tracking of face, hands and head orientation," in *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, (New York, NY, USA), pp. 140–146, ACM, 2003.

[24] Otsu, N., "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, pp. 62–66, Jan. 1979.

[25] Pavlovic, V., Sharma, R., and Huang, T., "Visual interpretation of hand gestures for human-computer interaction: a review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, pp. 677 –695, Jul. 1997.

[26] Rokade, U., Doye, D., and Kokare, M., "Hand gesture recognition using object based key frame selection," in *Digital Image Processing, 2009 International Conference on*, pp. 288 –291, 2009.

[27] Saman, G., Sajjad, A., Hameed, A., and Shah, A., "Visual sign language interpretation using spatial temporal neural processing," in *Multitopic Conference, 2006. INMIC '06. IEEE*, pp. 128 –133, 2006.

[28] Sangi, M. and Jahed, M., "A fast 3d hand model reconstruction by stereo vision system," in *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, vol. 5, pp. 545 –549, 2010.

[29] Segen, J. and Kumar, S., "Fast and accurate 3d gesture recognition interface," in *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition-Volume 1*, (Washington, DC, USA), p. 86, IEEE Computer Society, 1998.

[30] Stauffer, C. and Eric, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 747–757, Aug. 2000.

[31] Tanimoto, T. and Hoshino, K., "Real time posture estimation of human hand for robot hand interface," in *Universal Communication, 2008. ISUC '08. Second International Symposium on*, pp. 303 –308, 2008.

[32] Tosas, M., *Visual Articulated Hand Tracking for Interactive Surfaces*. PhD dissertation, University of Nottingham, Dec. 2006.

[33] Toyama, K., Krumm, J., Brumitt, B., and Meyers, B., "Wallflower: principles and practice of background maintenance," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 255 –261 vol.1, 1999.

[34] Viola, P. and Jones, M., "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–511 – I–518 vol.1, 2001.

[35] Zaletelj, J., Perhavc, J., and Tasic, J., "Vision-based human-computer interface using hand gestures," in *Image Analysis for Multimedia Interactive Services, 2007. WIAMIS '07. Eighth International Workshop on*, pp. 41 –41, 2007.